PAM & PAL: Policy-Aware Virtual Machine Migration and Placement in Dynamic Cloud Data Centers

Hugo Flores, Vincent Tran, and Bin Tang
Department of Computer Science
California State University Dominguez Hills, Carson, CA 90747, USA
Email: {hflores27,vtran42}@toromail.csudh.edu, btang@csudh.edu

***Abstract*—We focus on policy-aware data centers (PADCs), wherein virtual machine (VM) traffic traverses a sequence of middleboxes (MBs) for security and performance purposes, and propose two new VM placement and migration problems. We first study PAL: policy-aware virtual machine placement. Given a PADC with a data center policy that communicating VM pairs must satisfy, the goal of PAL is to place the VMs into the PADC to minimize their total communication cost. Due to dynamic traffic loads in PADCs, however, above VM placement may no longer be optimal after some time. We then study PAM: policy-aware virtual machine migration. Given an existing VM placement in the PADC and dynamic traffic rates among communicating VMs, PAM migrates VMs in order to minimize the total cost of migration and communication of the VM pairs. We design optimal, approximation, and heuristic *policy-aware* VM placement and migration algorithms. Our experiments show that i) VM migration is an effective technique, reducing total communication cost of VM pairs by around 30%, ii) our PAL algorithms outperform state-of-the-art VM placement algorithm that is oblivious to data center policies by 40-50%, and iii) our PAM algorithms outperform the *only* existing policy-aware VM migration scheme by 20-40%.**

***Index Terms*—Policy-Aware Data Centers, Virtual Machine Placement, Virtual Machine Migration, Algorithms**

I. Introduction

Recently, Middleboxes (MBs) [8], such as firewalls, load balancers, and network address translators, are introduced into cloud data centers to improve security and performances of virtual machine (VM) applications [37]. In particular, *data center policies* (or *service function chaining*) are established that require VM traffic to traverse a chain of MBs [24], [28], [44]. Fig. 1(a) shows such an example. Traffic generated at VM vm_1 goes through a firewall, a load balancer, and a cache proxy before arriving at VM $vm_1^{'}$. In doing so, this policy filters out malicious traffic, diverts trusted VM traffic to avoid network congestion, and finally caches the content to share with other cloud users. We refer to cloud data centers that implement such policies as *policy-aware data centers (PADCs)*. Fig. 1(b) shows a small PADC that implements the same data center policy in Fig. 1(a). A firewall, a load balancer, and a cache proxy are installed on switches s_2, s_3, and s_4, respectively. There are four physical machines (PMs), pm_1, pm_2, pm_3, and pm_4, each can store one VM due to its limited *cloud resources* (i.e., CPU, memory, and disk I/O).

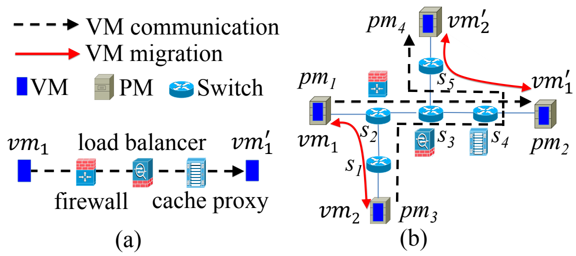


Fig. 1. (a) A data center policy. (b) A PADC example.

In this paper, we identify, formulate, and solve two new VM placement and migration problems in PADCs. Measurements from Facebook and other production data centers show that traffic loads (i.e., transmission rates) of communicating VM applications are highly diverse and dynamic [7], [34]. With an explosive growth of communicating VM applications and their ensued network traffic, *network resources* such as bandwidths and switches' processing capabilities become a performance bottleneck. Thus it is important to design resource-efficient VM placement and migration algorithms for dynamic cloud data centers. This is especially crucial for PADCs – as the VM communication must go through a sequence of MBs, it generates more network traffic and consumes more network resources compared to traditional cloud data centers.

Initially, when new cloud applications are submitted and created as VMs in PADCs, it needs to decide how to place them for resource-efficient communication. We refer to this problem as PAL: *policy-aware VM placement in PADCs*. Given a PADC with a data center policy that communicating VM pairs must satisfy, PAL studies how to place the VMs into the PMs to minimize their total communication cost while satisfying resource constraints of PMs. Fig. 1(b) shows that two VM pairs $(vm_1, vm_1^{'})$ and $(vm_2, vm_2^{'})$, with the traffic load of the former much larger than that of the latter, are to be placed. To reduce network traffic and communication delay, it is preferred that vm_1 and $vm_1^{'}$ communicates along a route that is "shorter" than that of vm_2 and $vm_2^{'}$. One solution is to place vm_1 and $vm_1^{'}$ on pm_1 and pm_2, and vm_2 and $vm_2^{'}$ on pm_3 and pm_4, respectively, with their respective policy-aware communication routes shown in black dashed lines.

However, such initial VM placement may no longer be optimal due to dynamic traffic loads in PADCs. Take Fig. 1(b) for example. If the traffic load of $(vm_2, vm_2^{'})$ emerges as much larger than that of $(vm_1, vm_1^{'})$, the aforesaid VM placement becomes inefficient – as vm_2 communicates with $vm_2^{'}$ via a route much longer than that of $(vm_1, vm_1^{'})$, it generates more network traffic and consumes much of the network bandwidths. We observe that migrating VMs from one PM to another might be an effective technique to tackle such dynamic VM traffic. For example, in Fig. 1(b), it might be a good idea to "swap" these two VM pairs using VM migrations shown in red solid lines. In particular, given an existing placement of communicating VM pairs with dynamic traffic rates, a data center policy that they must satisfy, the goal is to migrate VMs to minimize the total cost of migration and communication of all the VM pairs. We refer to this problem as PAM: *policy-aware VM migration in PADCs*.

Consider that VM migration incurs traffic overhead, and that a large scale PADC typically has tens of thousands of PMs, as well as hundreds of thousands of communicating VMs with wide range of changing traffic rates, how to effectively place and migrate VMs in PADCs to achieve optimal network resource utilization becomes challenging problem. In this paper we addresses this challenge and design optimal, approximation, and heuristic *policy-aware* VM placement and migration algorithms to solve PAL and PAM. The PAM & PAL duo potentially achieves ideal resource utilization for a PADC's lifetime - after the PAL algorithms create the initial VM placement to optimize a PADC's cloud resource utilization, the PAM algorithms can then be executed periodically to optimize a PADC's network resource utilization in the event of dynamic VM traffic. To the extent of our knowledge, PAL and PAM are new problems that have not been studied before.

Using traffic patterns and flow characteristics found in production data centers, we show that VM migration reduces the total communication costs of VM pairs by about 30%, demonstrating that it is an effective technique to alleviate dynamic VM traffic in PADCs. We also show our *policy-aware* algorithms outperform the *only* existing policy-aware VM migration algorithm [13] by 20-40%, and the state-of-the-art VM placement algorithm that is oblivious to data center policies [31] by 40-50%.

II. RELATED WORK

Service function chaining (SFC) has been a very active research in recent years. It mainly focused on virtual network functions (VNFs) (i.e., virtual MBs) with their implementation and realization [33], [17], [19], [44], [40], VNF placements [25], [32], VNF migrations [15], [29], and other issues such as availability [16], flow control [35], and finding shortest SFC [18]. However, given that many network functions still rely on dedicated hardware as virtual MBs cannot yet match the performance of hardware MBs, and much existence of hardware MBs in enterprise networks [23], [36], we consider hardware MBs, which cannot be easily moved around. However, our proposed approach still works the VNF scenarios.

There is vast amount of literature of VM migration in cloud data centers [38], [43], [6], [39], [41], [14]. In particular, Shrivastava et al. [38] proposed an application-aware VM migration that minimizes data center network traffic while considering the combined effects of application dependencies and network topology. Zhang et al. [43] analyzed how much bandwidth is required to guarantee the total migration time and downtime of a live VM migration. Wang et al. [39] studied how to migrate multiple VMs at the same time with available bandwidth, and designed a fully polynomial time approximation algorithm. Cui et al. [14] assumed that data center topologies are adaptive with reconfigurable wireless links or optical circuit switches, and proposed VM migration algorithm with constant approximation ratio.

VM migration studied in this paper, however, differs from aforesaid work in both goals and models. While existing VM migration work achieved various objectives such as server consolidation and energy efficiency, load balancing and fault tolerance, our work focuses on the dynamic communication traffic rates existing among VMs. Besides, none of the above work considered data center policies, thus falling short of achieving performance and security guarantees brought about by various of MBs deployed inside PADCs.

Meng et al. [31] designed one of the first *policy-oblivious* VM placement algorithms. It is traffic-aware in that it assigns VMs with large communications to the same PMs or PMs in close proximity. As PAL is the first to study *policy-aware* VM placement thus no closely related work to compare with, we compare our work with this traffic-aware VM placement. Alicherry and Lakshman [5] designed optimal and approximation algorithms that place VMs to minimize data access latencies. Li et al. [27] studied the VM placement to reduce the data center network cost as well as the cost utilizating PMs. Again, they are policy-oblivious thus do not achieve performance and security guarantees brought by PADCs.

PACE [26] was one of the first to study *policy-aware* VM placement. However, it only considers one type of MB thus does not study data center policy addressed in this paper. The only closely related work to ours is by Cui et al. [13], [12]. They proposed PLAN, the first policy-aware VM migration scheme for all-pair VM communications. As the problem is NP-hard, they provided heuristic algorithms for ordered-policies. In contrast, we focus on pairwise VM communication as the east-west traffic within the data center accounts for 75.4 percent of data center traffic [1], and most east-west traffic in cloud data centers is pairwise [31]. We design optimal, approximate, and heuristic algorithms for both ordered- and unordered-policies and show they all outperform PLAN.

III. SYSTEM MODELS

Network Model. We use fat tree [4] to illustrate the problems and their algorithmic solutions. However, as the problems are applicable to any data center topology, we model a PADC as an undirected general graph $G(V, E)$. $V = V_p \cup V_s$ is a set of PMs $V_p = \{pm_1, pm_2, ..., pm_{|V_p|}\}$ and a set of switches V_s. E is a set of edges, each connecting either one switch to another switch or a switch to a PM. Fig. 2 shows a $k = 4$ PADC where k is the number of ports each switch has.

There are a set of n MBs $\mathcal{M} = \{mb_1, mb_2, ..., mb_n\}$ inside the PADC, with mb_j installed at switch $sw_j \in V_s$. We adopt the *bump-off-the-wire* design [24], which uses a policy-aware switching layer to explicitly redirect traffic to off-path MBs. Fig. 2 shows three MBs mb_1, mb_2 and mb_3 installed using this design. As a switch and its attached MBs are connected by high-speed optical fibers, the delay between them is negligible compared to that among switches and PMs [21].

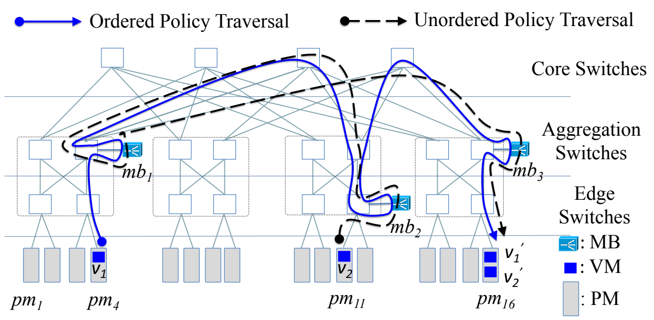


Fig. 2. A PADC with 16 PMs: pm_1, pm_2, ..., and pm_{16}, 3 MBs: mb_1, mb_2, and mb_3, and two VM pairs: $(v_1, v_1^{'})$ and $(v_2, v_2^{'})$. ● and ▶ indicate source and destination VM respectively.

There are l pairs of communicating VMs $\mathcal{P} = \{(v_1, v_1^{'}), (v_2, v_2^{'}), ..., (v_l, v_l^{'})\}$ that are already placed into the PMs. For any VM pair $(v_i, v_i^{'})$, $1 \leq i \leq l$, v_i and $v_i^{'}$ are referred to as its *source* and *destination VM* respectively. Denote the traffic rate or transmission rate of $(v_i, v_i^{'})$ as λ_i, and the *traffic rate vector* as $\overrightarrow{\lambda} = \langle \lambda_1, \lambda_2, ..., \lambda_l \rangle$. In a dynamic PADC, $\overrightarrow{\lambda}$ is not a constant vector as the VM traffic rates change over time. In Fig. 2, there are two VM pairs: $(v_1, v_1^{'})$ and $(v_2, v_2^{'})$, with $\overrightarrow{\lambda} = \langle 100, 1 \rangle$.

Let $\mathcal{V} = \{v_1, v_1^{'}, v_2, v_2^{'}, ..., v_l, v_l^{'}\}$. We assume that it needs one unit of cloud resource to create and execute a VM in $\mathcal{V}$ and leave the more general case with varying resource demands as future work. Here the *resource* is an aggregated characterization of a PM's hardware resources such as CPUs, memories, and disk I/O. Denote the *resource capacity* of PM pm_i as rc_i, meaning that pm_i has rc_i *resource slots*. As there are $2 \cdot l$ VMs and each needs one resource slot, it must be that $\sum_{i=1}^{|V_p|} rc_i \geq 2 \cdot l$. Table I shows all the notations.

Cost Model. We model the VM communication and migration cost as either the delay or energy cost of the network traffic inside PADCs. We adopt a *topology-aware* model [31] and define the *communication cost* of any VM pair as the number of network links its traffic traverses multiplied by its traffic rate (however, our problems and solutions still hold for that different edges have different costs). The *migration cost* of migrating any VM v from PM i to PM j is $\mu \cdot c(i, j)$. Here, $c(i, j)$ is the minimum number of hops between any PM (or switch) i and j, and μ is a *migration coefficient* that depends on VM sizes and available network bandwidths.

Justifications. Our VM migration model is different from the one adopted by most existing literature. Mann et al. [30] focused on *pre-copy* [9], one of the original live VM migration techniques, and modeled the cost of migrating a VM v as $M_s \cdot \frac{1-(P_r/B_a)^{n+1}}{1-(P_r/B_a)}$. Here M_s is the image size of v, P_r is its page dirty rate, B_a is the available bandwidth, and n is number of pre-copy phases. They suggested the migration cost be a constant quantity measured by the hypervisor. In contrast, by acknowledging network delay or energy consumption incurred by VM migration traffic, our topology-aware model is more conducive to designing VM migration algorithms for a large-scale and dynamical traffic environment targeted by this paper.

TABLE I
NOTATION SUMMARY

Notation	Description
V_p	$V_p = \{pm_1, pm_2, ..., pm_{\vert V_p \vert}\}$ is the set of $\vert V_p \vert$ PMs
V_s	Set of switches in a PADC
$\mathcal{M}$	$\mathcal{M} = \{mb_1, mb_2, ..., mb_n\}$ is the set of n MBs
$\mathcal{P}$	$\mathcal{P} = \{(v_i, v_i^{'}), ..., (v_l, v_l^{'})\}$ is the set of l VM pairs
$\mathcal{V}$	$\mathcal{V} = \{v_1, ..., v_l, v_1^{'}, ..., v_l^{'}\}$
λ_i	Traffic rate between v_i and $v_i^{'}$, $1 \leq i \leq l$
$\overrightarrow{\lambda}$	$\overrightarrow{\lambda} = \langle \lambda_1, \lambda_2, ..., \lambda_l \rangle$
rc_i	Resource capacity of PM pm_i, $1 \leq i \leq \vert V_p \vert$
sw_j	Switch where mb_j is installed, $1 \leq j \leq n$
$c(i, j)$	Communication cost between PMs (or switches) i and j
$p(v)$	PM where the VM v is placed under VM placement p
π^i	Order at which $(v_i, v_i^{'})$ visits MBs in unordered policy
$\overrightarrow{\pi}$	$\overrightarrow{\pi} = \langle \pi^1, \pi^2, ..., \pi^l \rangle$
$C_c(p)$	Total communication cost under p in ordered policy
$C_c(p, \overrightarrow{\pi})$	Total communication cost under p in unordered policy
μ	Migration coefficient
$m(v)$	PM where the VM v migrates to under VM migration m
$C_m(m)$	Total migration cost with migration m
$C_c(m)$	Total communication cost with migration m
$C_t(m)$	Total migra. and comm. cost with m in ordered policy
$C_t(m, \overrightarrow{\pi})$	Total migra. and comm. cost with m in unordered policy

Data Center Policies. Depending on the application requirements, some policies require that the VM traffic to go through the MBs in a strict order. We refer to such policies as *ordered policies* and denote them as $(mb_1, mb_2, ..., mb_n)$. On the other hand, as MB functions are mostly independent from each other, some data center policies are considered satisfied as long as all its MBs are visited by the VM traffic. We refer to such policies as *unordered policies* and denote them as $\{mb_1, mb_2, ..., mb_n\}$. In Fig. 2, $(v_1, v_1^{'})$ traverses MBs under ordered-policy (mb_1, mb_2, mb_3), resulting in communication cost of $100 \times 10 = 1000$ (solid blue line), $(v_2, v_2^{'})$ traverses MBs under unordered-policy $\{mb_1, mb_2, mb_3\}$, resulting in communication cost of $1 \times 8 = 8$ (dashed black line).

We refer to the first (and last) visited MB in a policy as *ingress (and egress) MB*, and the switch where the ingress (and egress) MB is installed as *ingress (and egress) switch*. For ordered policy, the ingress switch is sw_1 and egress switch is sw_n. For unordered policy, it needs to find out sw_1 and sw_n as well as the MB sequence along which VM pair communicates. As shown in Fig. 1(a), one data center policy is generally sufficient to serve both security and performance purposes. We thus assume there is one data center policy (ordered or unordered) in a PDDC at a time. We adopt FlowTags [17], a SDN architecture that provides consistent policy enforcement during VM migration by adding tags in packet headers.

EXAMPLE 1: Fig. 3 shows a $k = 2$ linear fat tree PADC with two PMs: pm_1 and pm_2. Each PM has two resource slots; the four of them are $\{rs_1, rs_2, rs_3, rs_4\}$. Two MBs, mb_1

and mb_2, are installed on edge switch sw_1 and aggregation switch sw_2, respectively. There are two VM pairs $(v_1, v_1^{'})$ and $(v_2, v_2^{'})$, v_1 and v_2 are placed on pm_1 while $v_1^{'}$ and $v_2{'}$ on pm_2. $\overrightarrow{\lambda} = \langle 100, 1\rangle$ and $\mu = 1$. Before migration, the total communication cost in Fig. 3(a) is 606 under both (mb_1, mb_2) and $\{mb_1, mb_2\}$. By migrating $v_1^{'}$ to pm_1 and v_2 to pm_2 with migration cost of 12 (solid red line in Fig. 3(a)), the total communication cost (dotted and dashed black lines in Fig. 3(b)) becomes 410, a 30% of total cost reduction. We show this migration is indeed optimal in Section V-A. □

IV. PAL: Policy-Aware VM Placement in PADCs

A. Ordered Policy.

1) Problem Formulation: Under ordered policy, for any VM pair communication, the ingress switch is always sw_1 and the egress switch is always sw_n. Given a VM placement function p, denote the *total communication cost* of all the l VM pairs under p as $C_c(p)$. We have $C_c(p) =$

$$= \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{n-1} c\big(sw_j, sw_{j+1}\big) + \sum_{i=1}^{l} \lambda_i \cdot \Big(c\big(p(v_i), sw_1\big) + c\big(sw_n, p(v_i^{'})\big)\Big). \quad (1)$$

The objective of PAL is to find a VM placement p to minimize $C_c(p)$ while satisfying resource constraint of PMs: $|\{v \in \mathcal{V} | p(v) = i\}| \le rc_i, \forall i \in V_p$. As the first term on the r.h.s. of Eq. 1 is fixed under ordered policy, we only need to minimize the second term. Below we design an optimal and efficient algorithm to solve PAL.

2) VM Placement Algorithm for Ordered Policy: To save communication costs for VM pairs, the key is to find a set of resource slots close to the ingress (and egress) switch to place source (and destination) VMs. We give below definitions.

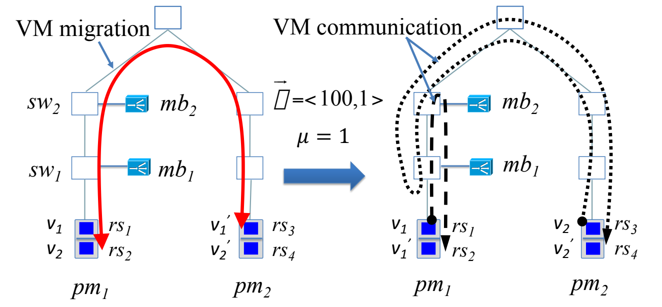
Definition 1: (Ingress/Egress Costs, Ingress/Egress Resource Sets, Optimal Ingress/Egress Sets) A resource slot rs's *ingress* (and *egress*) cost, denoted as $c_{in}(rs)$ (and $c_e(rs)$), is the cost between its belonged PM and the ingress switch sw_1 (and egress switch sw_n). Let $pm(rs)$ be the PM rs belongs to, $c_{in}(rs) = c\big(pm(rs), sw_1\big)$, $c_e(rs) = c\big(pm(rs), sw_n\big)$.

An *ingress* (and *egress*) *resource set* (IRS and ERS) is a set of l resource slots that store the l source (and destination) VMs. The *cost* of an IRS (and ERS) is the sum of the ingress (and egress) costs of its resource slots. A pair of IRS and ERS is *optimal*, denoted as (I^{opt}, E^{opt}), if the sum of their costs is the minimum among all pairs of IRS and ERS. □

I^{opt} and E^{opt} are structures that uniquely arise in PAL. Algo. 1 below finds such a pair (lines 1-22) and then places the l VM pairs (in non-ascending order of their traffic rates) into it (lines 23-30). Its time complexity is $O(|V_p|^2 \cdot \bar{m}^2)$, where $\bar{m}$ is the average resource capacity of a PM.

Algorithm 1: PAL Algorithm for Ordered Policy.
Input: A PADC with ordered policy $(mb_1, mb_2, ..., mb_n)$, VM pairs $\mathcal{P}$, $V_p = \{pm_i\}$, resource capacity rc_i.
Output: A placement p and the total comm. cost $C_c(p)$.
Notations: $sel(rs_i)$: true if rs_i is selected into either I^{opt} or E^{opt}, initially false for all resource slots.
$\mathcal{I}$ and $\mathcal{E}$: arrays of resource slots, each of size l.
i, j: indices for $\mathcal{I}$ and $\mathcal{E}$ respectively.
k: index for I^{opt} and E^{opt}.

1. $i = j = k = 1$, $C_c(p) = 0$, $p = \phi$;
2. Sort resource slots in non-descending order of their ingress and egress costs, store the top $2l$ in arrays $\mathcal{I}$ and $\mathcal{E}$;
3. **while** $(k \le l)$ // find optimal resource slots for $(v_k, v_k^{'})$
4. **if** $(sel[\mathcal{I}[i]] == true)$ i++;
5. **if** $(sel[\mathcal{E}[j]] == true)$ j++;
6. **if** $(\mathcal{I}[i] \neq \mathcal{E}[j]$) // both optimal resource slots are found
7. $I^{opt}[k] = \mathcal{I}[i]$, $E^{opt}[k] = \mathcal{E}[j]$;
8. $sel[\mathcal{I}[i]] = sel[\mathcal{E}[j]] = true$;
9. i++, j++;
10. **else** // one found, now find the other
11. **if** $(c_{in}(\mathcal{I}[i]) + c_e(\mathcal{E}[j+1]) \le c_{in}(\mathcal{I}[i+1]) + c_e(\mathcal{E}[j]))$
12. $I^{opt}[k] = \mathcal{I}[i]$, $E^{opt}[k] = \mathcal{E}[j+1]$;
13. $sel[\mathcal{I}[i]] = sel[\mathcal{E}[j+1]] = true$;
14. i++, j += 2;
15. **else**
16. $I^{opt}[k] = \mathcal{I}[i+1]$, $E^{opt}[k] = \mathcal{E}[j]$;
17. $sel[\mathcal{I}[i+1]] = sel[\mathcal{E}[j]] = true$;
18. i += 2, j++;
19. **end if;**
20. **end if;**
21. k++;
22. **end while;**
23. WLOG, $\lambda_1 \ge \lambda_2 ... \ge \lambda_l$;
24. **for** $(1 \le i \le l)$ // place VM pairs and calculate cost
25. Place v_i at resource slot $I^{opt}[i]$;
26. Place $v_i^{'}$ at resource slot $E^{opt}[i]$;
27. $p = p \cup \{(I^{opt}[i], E^{opt}[i])\}$;
28. $C_c(p)$ += $\lambda_i * \big(c_{in}(I^{opt}[i]) + c_e(E^{opt}[i])\big)$;
29. **end for;**
30. $C_c(p)$ += $\sum_{i=1}^{l} \lambda_i \sum_{j=1}^{n-1} c\big(sw_j, sw_{j+1}\big)$;
31. **RETURN** p and $C_c(p)$.



(a) Before migration (b) After migration

Fig. 3. VM migration achieved 30% of total cost reduction in a linear PADC.

EXAMPLE 2: Fig. 4(a) shows how Algo. 1 could place the two VM pairs $(v_1, v_1^{'})$ and $(v_2, v_2^{'})$ into the same PADC in Fig. 3. It gives $\mathcal{I} = \mathcal{E} = \{rs_1, rs_2, rs_3, rs_4\}$, from which it computes $I^{opt} = \{rs_1, rs_3\}$ and $E^{opt} = \{rs_2, rs_4\}$. It thus places v_1 and $v_1^{'}$ in pm_1 and v_2 and $v_2{'}$ in pm_2 with total communication cost of $100 \cdot 4 + 1 \cdot 10 = 410$. □

Theorem 1: Algo. 1 finds the VM placement that mini-

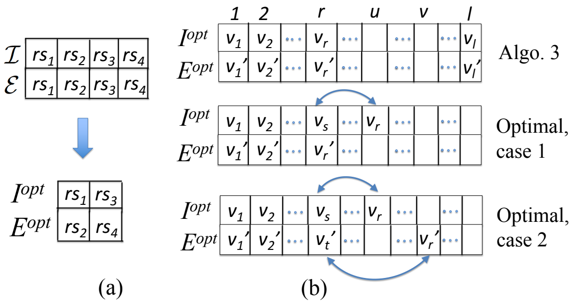


Fig. 4. (a) A working example and (b) optimality proof for Algo. 1.

mizes total communication cost for the l VM pairs.

Proof Sketch. First, we prove by induction that (I^{opt}, E^{opt}) computed in Algo. 1 (lines 1-22) is a pair of optimal IRS and ERS. Second, we prove by contradiction that the VM placement on I^{opt} and E^{opt} in Algo. 1 (lines 23-30) yields minimum total communication cost. Assume that Algo. 1 is not optimal and that r, $1 \leq r \leq l$, is the smallest index at which $I^{opt}[r]$ or $E^{opt}[r]$ store different pair of VMs for Algo. 1 and Optimal. Two cases are shown in Fig. 4(b). Case 1: one of the two resource slots, $I^{opt}[r]$ or $E^{opt}[r]$, stores different VMs. Case 2: both slots store different VMs. In both cases, we are able to swap VMs in Optimal (blue curved arrow lines) to further reduce its cost, due to $\lambda_1 \geq \lambda_2 ... \geq \lambda_l$. ∎

B. Unordered Policy.

1) Problem Formulation.: In unordered policy, besides a VM placement function p, PAL needs to find the order at which *each* VM pair visits the MBs. For $(v_i, v_i^{'})$ we define an MB *traversal function* $\pi^i : [1, 2, ..., n] \to [1, 2, ..., n]$, a permutation function indicating the j^{th} MB that $(v_i, v_i^{'})$ visits is $mb_{\pi^i(j)}$. Given p and π^i, denote $(v_i, v_i^{'})$'s communication cost as c_i^{p,π^i}. Then $c_i^{p,\pi^i} = \lambda_i \cdot c\Big(p(v_i), sw(\pi^i(1))\Big) + \lambda_i \cdot \sum_{j=1}^{n-1} c\Big(sw\big(\pi^i(j)\big), sw\big(\pi^i(j+1)\big)\Big) + \lambda_i \cdot c\Big(sw\big(\pi^i(n)\big), p(v_i^{'})\Big)$. Let $\overrightarrow{\pi} = \langle \pi^1, \pi^2, ..., \pi^l \rangle$. The objective of PAL under unordered policy is to minimize *total communication cost* $C_c(p, \overrightarrow{\pi}) = \sum_{i=1}^{l} c_i^{p,\pi^i}$ while satisfying $|\{v \in \mathcal{V} | p(v) = i\}| \leq rc_i, \forall i \in V_p$. Below we show that PAL is NP-hard even for one pair of VMs. We then propose an approximation algorithm for this special case that yields total cost at most twice of the optimal.

Theorem 2: Under unordered policy, PAL is NP-hard even for one pair of VMs $(v_1, v_1^{'}$ (i.e., $l = 1$).

Proof: We reduce *s-t traveling salesman path problem* (TSPP) [22], which is NP-hard, to this problem. Given a complete undirected graph $K = (V_K, E_K)$ with edge cost $d : E_K \to \mathbb{R}^+$ and a pair of pre-specified vertices $s, t \in V_K$, TSPP finds a shortest *Hamiltonian path* that starts at s, visits each vertex exactly once, and ends at t. d satisfies *triangle inequality*, i.e., $d(u, v) \leq d(u, v) + d(v, w)$ for all $u, v, w \in V_K$, When $s = t$, TSPP becomes well-known *traveling salesman problem* (TSP) [11], which finds a shortest *Hamiltonian cycle*.

Given VM pair $(v_1, v_1^{'})$ and an instance of PADC graph $G(V = V_p \cup V_s, E)$, we construct $|V_p| \cdot (|V_p| + 1)/2$ instances of complete graphs $K^{i,j} = (V_K^{i,j}, E_K^{i,j})$, $1 \leq i \leq |V_p|$, $i \leq j \leq |V_p|$. Here, $V_K^{i,j} = \{pm_i, pm_j, sw_1, sw_2, ...sw_n\}$ and for $(u, v) \in E_K^{i,j}$, its cost $d(u, v)$ is the cost of the shortest path connecting u and v in G. Now, if $K^{a,b}$ has a shortest Hamiltonian path whose cost is the minimum among the shortest Hamiltonian paths in all the instances, then placing v_1 to pm_a and $v_1^{'}$ to pm_b must give the minimum communication cost for $(v_1, v_1^{'})$ in G, and vice versa. ∎

2) VM Placement Algorithm for Unordered Policy:

Definition 2: (**Optimal Policy Route (OPR).**) In a PADC graph, a *policy route* of any pair of PMs (pm_i, pm_j) is a path or walk starting pm_i, visiting all the n MBs at least once, and ending at pm_j. An OPR of (pm_i, pm_j) gives the minimum cost, denoted as $opr(i, j)$, among all its policy routes. □

OPR of (pm_i, pm_j) is essentially the shortest *s-t* Hamiltonian path [22] with $s = pm_i$ and $t = pm_j$ in a complete graph of pm_i, pm_j and all MBs (when $pm_i = pm_j$, it is a Hamiltonian cycle). The existing algorithm achieves approximation ratio of $\frac{5}{3}$ and takes $O(n^3)$ [22], where n is the number of MBs. Below we instead propose another $O(n^3)$ but simpler algorithm to compute a policy route for (pm_i, pm_j) and show it has an approximation ratio of 2.

Algorithm 2: Compute A Policy Route for (pm_i, pm_j).
Input: A PADC graph G, a pair of PMs (pm_i, pm_j),
and an unordered policy $\{mb_1, mb_2, ..., mb_n\}$.
Output: $pr(i, j)$, cost of a policy route for (pm_i, pm_j).

1. $V_K^{i,j} = \{pm_i, pm_j, sw_1, sw_2, ...sw_n\}$;
2. Construct complete graph $K^{i,j} = (V_K^{i,j}, E_K^{i,j})$. For edge $(u, v) \in E_K^{i,j}$, its cost $d(u, v)$ is the cost of the shortest path connecting u and v in G;
3. Compute a minimum spanning tree MST for $K^{i,j}$;
4. Compute a walk W from pm_i to pm_j on MST that visits all vertices in MST using each edge *at most twice*. Let the cost of W be $pr(i, j)$;
5. **RETURN** $pr(i, j)$.

Using Algo. 2, we show in Fig. 5 (blue dashed lines) all three possible policy routes in the linear PADC of Fig. 3.

Lemma 1: $pr(i, j) \leq 2 \cdot opr(i, j)$, $\forall pm_i, pm_j \in V_p$.

Proof: Denote the cost of the MST computed in line 3 as c(MST), c(MST) $\leq opr(i, j)$. Since the walk W found in line 4 uses each edge of the MST at most twice, its cost $pr(i, j) \leq 2 \cdot c$(MST). Thus we have $pr(i, j) \leq 2 \cdot opr(i, j)$. ∎

Next we present our PAL algorithm Algo. 3. It first computes the policy routes for all the $|V_p| \cdot (|V_p| + 1)/2$ pair of PMs using Algo. 2 and orders them in the non-descending order of their costs (lines 1-8). It then places the VM pairs (in the non-ascending order of their traffic rates) onto the first available PM pair, and updates the total communication cost accordingly (lines 9-21). Running time of Algo. 3 is $O(|V_p|^2 \cdot n^3 + l)$.

Algorithm 3: PAL Algorithm for Unordered Policy.
Input: A PADC with unordered policy $\{mb_1, mb_2, ..., mb_m\}$,
VM pairs $\mathcal{P}$, $V_p = \{pm_i\}$, resource capacity rc_i.
Output: A placement p and the total comm. cost $C_c(p, \overrightarrow{\pi})$.
Notations: $avail(pm_i)$: available resource slots at pm_i.
X: all pairs of PMs with their SPR costs.

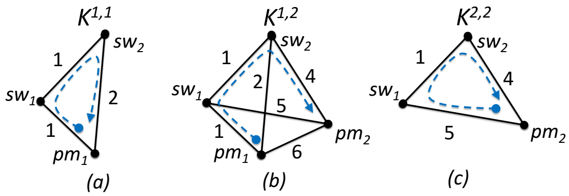


Fig. 5. Illustrating how Algo. 3 places $(v_1, v_1^{'})$ and $(v_1, v_1^{'})$ in the linear PADC of Fig. 3. The blue dashed lines show policy routes when (a) both v_i and $v_i^{'}$ ($i = 1$ or 2) are placed in pm_1, (b) v_i is in pm_1 and $v_i^{'}$ in pm_2, and (c) both v_i and $v_i^{'}$ are in pm_2.

$X = \phi$; $avail(pm_i) = rc_i, \forall pm_i \in V_p$;
for $(i = 1; i \leq |V_p|; i{+}{+})$
  for $(j = i; j \leq |V_p|; j{+}{+})$
    Compute $pr(i, j)$ using Algo. 2;
    $X = X \cup \{(i, j, pr(i, j))\}$;
  end for;
end for;
Sort X in non-descending order of $pr(i, j)$. Let X be $\{(s_1, t_1, pr(s_1, t_1)), (s_2, t_2, pr(s_2, t_2)), ...\}$;
$i = 1$, $j = 1$, $p = \phi$, $C_c(p, \overrightarrow{\pi}) = 0$;
WLOG, $\lambda_1 \geq \lambda_2 ... \geq \lambda_l$;
while $(i \leq l)$ // not all VM pairs are placed yet
  while $\big(avail(s_j) \geq 1 \wedge avail(t_j) \geq 1\big)$
    Place v_i at PM s_j, place $v_i^{'}$ at PM t_j;
    $p = p \cup \{(s_j, t_j)\}$;
    $C_c(p, \overrightarrow{\pi}) \mathrel{+}= \lambda_i * pr(s_j, t_j)$;
    $avail(s_j){-}{-}$, $avail(t_j){-}{-}$;
    $i{+}{+}$;
    if $(i > l)$ **break**;
  end while;
  $j{+}{+}$; // the next available PM pairs
end while;
RETURN p and $C_c(p, \overrightarrow{\pi})$.

EXAMPLE 3: Fig. 5 shows how Algo. 3 places $(v_1, v_1^{'})$ and $(v_2, v_2^{'})$, with $\overrightarrow{\lambda} = \langle 100, 1 \rangle$, in the linear PADC in Fig. 3. It computes $X = \{(1, 1, 4), (1, 2, 6), (2, 2, 10)\}$. Thus $(v_1, v_1^{'})$ is placed at pm_1 and communicate in blue dashed line in Fig. 5(a). As pm_1 is now full, $(v_2, v_2^{'})$ is placed at pm_2 and communicate shown in blue dashed line in Fig. 5(c). The total communication cost is $100 \cdot 4 + 1 \cdot 10 = 410$. □

Theorem 3: Algo. 3 achieves 2-approximation when $l = 1$.

Proof: Let the placement of $(v_1, v_1^{'})$ computed by Algo. 3 be (pm_a, pm_b). Let the optimal placement of $(v_1, v_1^{'})$ be $(pm_{a'}, pm_{b'})$ thus their optimal communication cost is $opr(a', b')$. From Lemma 1, we have $pr(a', b') \leq 2 \cdot opr(a', b')$. As the costs of all PM pairs are sorted in non-descending order in Algo. 3, $pr(a, b) \leq pr(a', b') \leq 2 \cdot opr(a', b')$. ■

V. PAM: Policy-Aware VM Migration in PADCs

A. Ordered Policy.

1) Problem Formulation: In PAM, the initial VM placement is given by a *placement function* $p : \mathcal{V} \to V_p$, indicating that VM $v \in \mathcal{V}$ is at PM $p(v) \in V_p$. The total communication cost of all the l VM pairs with placement p is thus $C_c(p)$ (Eq. 1). Next, define a VM *migration function* as $m : \mathcal{V} \to V_p$, meaning that VM v will be migrated from PM $p(v)$ to PM $m(v)$ ($m(v) = p(v)$ if v does not migrate). Let $C_m(m)$ denote the *total migration cost* of all the VM pairs; $C_m(m) = \mu \cdot \sum_{i=1}^{l} \Big(c\big(p(v_i), m(v_i)\big) + c\big(p(v_i^{'}), m(v_i^{'})\big) \Big)$. Let $C_t(m)$ denote the *total cost* of all pairs *after* VM migration m, which is the sum of the total migration cost and the total communication cost after VM migration m. Then $C_t(m) = C_m(m) + C_c(m)$

$$
\begin{aligned}
&= \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{n-1} c\big(sw_j, sw_{j+1}\big) \\
&+ \sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i), m(v_i)\big) + \lambda_i \cdot c\big(m(v_i), sw_1\big) \Big) \\
&+ \sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i^{'}), m(v_i^{'})\big) + \lambda_i \cdot c\big(sw_n, m(v_i^{'})\big) \Big).
\end{aligned} \tag{2}
$$

The objective of PAM is to find a VM migration m that minimizes $C_t(m)$ while satisfying resource constraint of PMs: $|\{v \in \mathcal{V} | m(v) = pm_i\}| \leq rc_i, \forall pm_i \in V_p$. As the first term on the right hand side in Eq. 2 is fixed under ordered policy, we only need to minimize the sum of the last two terms. Below we show that PAM under ordered policy is equivalent to minimum cost flow problem [3] in a properly transformed flow network.

2) Minimum Cost Flow (MCF) Problem: MCF is formally defined as follows. Let $G = (V, E)$ be a directed graph. Denote the capacity and cost of an edge $(u, v) \in E$ as $ca(u, v)$ and $d(u, v)$, respectively. The amount of supply from source node $s \in V$ equals the amount of demand at sink node $t \in V$. Denote a flow on edge (u, v) as $f(u, v)$, $f : E \to \mathbb{R}^+$. $f(u, v)$ is subject to (a) capacity constraint: $f(u, v) \leq ca(u, v), \forall (u, v) \in E$ and (b) flow conservation constraint: $\sum_{u \in V} f(u, v) = \sum_{u \in V} f(v, u)$, for each $v \in V \setminus \{s, t\}$. The goal of MCF is to find a flow function f such that the total cost of the flow $\Sigma_{(u,v) \in E} \big(d(u, v) \cdot f(u, v)\big)$ is minimized. MCF can be solved efficiently and optimally by many combinatorial algorithms [3]. In this paper, we adopt the scaling push-relabel algorithm proposed by Goldberg [20] as it works well over a wide range of problem classes. The algorithm has the time complexity of $O(A^2 \cdot B \cdot \log(A \cdot C))$, where A, B, and C are the number of nodes, number of edges, and maximum edge capacity in the flow network, respectively.

Transforming a PADC to a Flow Network. We transform a PADC $G(V, E)$ into a flow network $G'(V', E')$ following below five steps, as shown in Fig. 6(a).

Step I. $V' = \{s\} \cup \{t\} \cup \mathcal{V} \cup V_p$, where s is the source node and t is the sink node in the flow network.

Step II. $E' = \{(s, v)\} \cup \{(v, pm_j)\} \cup \{(pm_j, t)\}$, where $v \in \mathcal{V}$ and $pm_j \in V_p$. Note that it is a complete bipartite graph between $\mathcal{V}$ and V_p.

Step III. For each edge (s, v), set its capacity as 1 and cost as 0. For each edge (pm_j, t), set its capacity as rc_j, the resource capacity of pm_j, and cost as 0.

Step IV. For each edge (v_i, pm_j), set its capacity as 1 and cost as $\mu \cdot c\big(p(v_i), pm_j\big) + \lambda_i \cdot c\big(pm_j, sw_1\big)$. For each edge $(v_i^{'}, pm_j)$, set its capacity as 1 and cost as $\mu \cdot c\big(p(v_i^{'}), pm_j\big) + \lambda_i \cdot c\big(pm_j, sw_n\big)$.

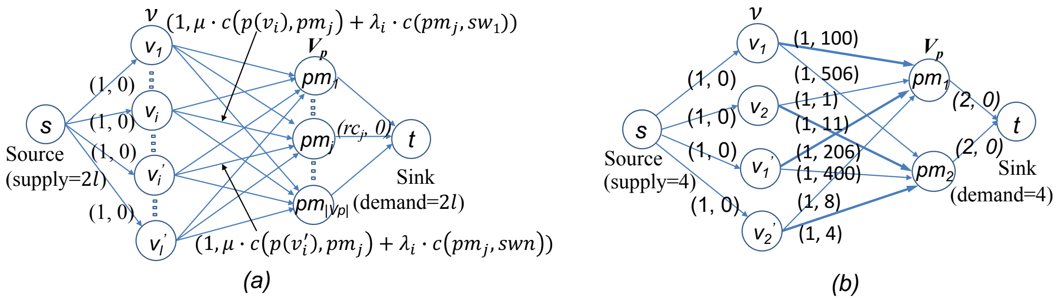


Fig. 6. (a) PAM under ordered policy is equivalent to MCF problem. (b) Graph transformation for the PADC in Fig. 3. Highlighted lines show the VM migrations resulted from the MCF algorithm.

Step V. Set the supply at s and the demand at t as $2l$.

Theorem 4: PAM in ordered policy is equivalent to MCF in $G'(V', E')$ thus can be solved optimally and efficiently.

Proof Sketch: Due to space constraint, we only give a high level sketch of the proof. By applying MCF algorithm upon the above flow network, it is able to achieve that a) every VM in the l VM pairs is assigned to exactly one resource slot in a PM while b) satisfying the resource capacity constraints of PMs and c) achieving the minimum total migration and communication cost for all the l VM pairs. ∎

Time Complexity. As the number of nodes, edges, and maximum edge capacity in $G'(V', E')$ are $\bar{m} \cdot |V_p|$, $\bar{m} \cdot |V_p|^2$, and $\bar{m}$ respectively, the MCF takes $O\big(\bar{m}^3 \cdot |V_p|^4 \cdot \log(\bar{m}^2 \cdot |V_p|)\big)$.

EXAMPLE 4: Fig 6(b) illustrates how above transformation and MCF work for the same PADC in Fig. 3. MCF gives that v_1 and $v_1^{'}$ migrate to pm_1, v_2 and $v_2^{'}$ migrate to pm_2, with total cost of 100+11+206+4+101=422. Here 101 is the total communication cost between ingress switch sw_1 and egress switch sw_2. This migration reduces the total cost of 606 before migration by 30%. Note that as v_1 is initially located at pm_1 and $v_2^{'}$ at pm_2, only $v_1^{'}$ and v_2 actually migrate. □

3) State-of-the-Art VM Migration Scheme: Cui et al. [13] proposed a policy-aware VM management scheme named PLAN. The core concept of PLAN is *utility* of a VM migration. It is defined as a VM's communication cost reduction due to migration minus its migration cost (Definition 1, [13]). The goal of PLAN is to find a migration scheme that maximizes the *total utility* of migrating all the VMs. PLAN is a greedy algorithm (Algorithm 1, [13]) that works in rounds. In each round, it computes that which VM is migrated to which PM with available resources, such that the utility of this migration is the maximum among all the VMs that have not been migrated. This continues until all the VMs are migrated, or no more VM migration gives any positive utility.

PLAN is however a heuristic algorithm that does not offer any performance guarantee. We prove in Lemma 2 below that its goal of maximizing the total utilities is equivalent to our goal of minimizing total communication and migration cost in PAM, thus we can compare our algorithms with PLAN.

Lemma 2: Minimizing total cost $C_t(m)$ in PAM is equivalent to maximizing total utility in PLAN.

Proof: Denote the utility of migrating VM v as $u(v)$. Under migration function m, the *utility of migrating* v_i from its current PM $p(v_i)$ to another PM $m(v_i)$ is the reduction of its communication cost to the ingress switch minus the incurred migration cost. Thus $u(v_i) = \lambda_i \cdot \big(c(p(v_i), sw_1) - c(m(v_i), sw_1)\big) - \mu \cdot c\big(p(v_i), m(v_i)\big)$. Similarly, $u(v_i^{'}) = \lambda_i \cdot \big(c(p(v_i^{'}), sw_n) - c(m(v_i^{'}), sw_n)\big) - \mu \cdot c\big(p(v_i^{'}), m(v_i^{'})\big)$.

Given a p and a $\overrightarrow{\lambda}$, the total communication cost of the VMs $C_c(p)$ can be computed using Eq. 1. Thus minimizing $C_t(m)$ is equivalent to maximizing $C_c(p) - C_t(m) \overset{\text{Eqs. 1,2}}{=} \sum_{i=1}^{l} \lambda_i \cdot \Big(c\big(p(v_i), sw_1\big) + c\big(sw_n, p(v_i^{'})\big) - c\big(m(v_i), sw_1\big) - c\big(sw_n, m(v_i^{'})\big)\Big) - \mu \cdot \sum_{v \in \mathcal{V}} c\big(p(v), m(v)\big) = \sum_{i=1}^{l} \big(u(v_i) + u(v_i^{'})\big)$, which is the *total utility* of migrating all the VMs. ∎

B. Unordered Policy.

1) Problem Formulation: Under unordered policy, besides a VM *migration function* $m : \mathcal{V} \to V_p$, it defines for each VM pair $(v_i, v_i^{'})$ an MB *traversal function* $\pi^i : [1, 2, ..., n] \to [1, 2, ..., n]$. π^i is a permutation function indicating that after VM migration, the j^{th} MB that $(v_i, v_i^{'})$ visits is $mb_{\pi^i(j)}$. Let $\overrightarrow{\pi} = \langle \pi^1, \pi^2, ..., \pi^l \rangle$ and let $C_t(m, \overrightarrow{\pi})$ denote the *total cost* of all the VM pairs with m and $\overrightarrow{\pi}$. Then $C_t(m, \overrightarrow{\pi}) =$

$$\sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i), m(v_i)\big) + \mu \cdot c\big(p(v_i^{'}), m(v_i^{'})\big)\Big) + \sum_{i=1}^{l} \lambda_i \cdot \Bigg(\sum_{j=1}^{n-1} c\Big(sw\big(\pi^i(j)\big), sw\big(\pi^i(j+1)\big)\Big) + c\Big(m(v_i), sw\big(\pi^i(1)\big)\Big) + c\Big(sw\big(\pi^i(n)\big), m(v_i^{'})\Big)\Bigg). \quad (3)$$

The first and second terms in Eq. 3 are the total migration cost and total communication cost respectively. The objective of PAM under unordered policy is to find an m and a $\overrightarrow{\pi}$ to minimize $C_t(m, \overrightarrow{\pi})$ while satisfying resource constraints of PMs: $|\{v \in \mathcal{V} | m(v) = i\}| \leq rc_i, \forall i \in V_p$.

2) VM Migration Algorithm for Unordered Policy: Algo. 4 below first computes costs for all the $|V_p| \cdot (|V_p| + 1)/2$ policy routes (lines 2-6). Then for each VM pair (in the non-ascending order of their traffic rates), it finds a PM pair to migrate to, such that the resulted cost for this VM pair is the minimum among all the unassigned VM pairs in this round (lines 7-23). After the entire migration scheme m is computed, it finally migrates the VMs and returns the total cost (lines 24 and 25). Its takes $O(|V_p|^2 \cdot (n^3 + l))$.

Algorithm 4: PAM Algorithm for Unordered Policy.

Input: A PADC with unordered policy $\{mb_1, mb_2, ..., mb_n\}$,

$V_p = \{pm_i\}$, resource capacity rc_i, VM pair placement p.
Output: A migration scheme m and the total cost $C_t(m, \overrightarrow{\pi})$.
Notations: i, j: indices for PM pairs; k: index for VM pairs.
$c_{i,j}$: the total cost of a VM pair if its source VM is migrated to $pm(i)$ and destination VM to $pm(j)$.
a, b: indices of a PM pair that gives minimum total cost.
$avail(pm_i)$: number of available slots at pm_i, initially rc_i.

$m = \phi$, $C_t(m, \overrightarrow{\pi}) = 0$, $k = 1$, $\lambda_1 \geq \lambda_2 ... \geq \lambda_l$;
for $(i = 1; i \leq |V_p|; i{+}{+})$
  for $(j = i; j \leq |V_p|; j{+}{+})$
    Compute $pr(i, j)$ using Algo. 2;
  end for;
end for;
while $(k \leq l)$ // find PM pair for VM pair $(v_k, v_k^{'})$
  $c_{min} = \infty$; // minimum total cost for $(v_k, v_k^{'})$
  for $(i = 1; i \leq |V_p|; i{+}{+})$
    if $(avail(pm_i) == 0)$ **continue**; // pm_i is full
    for $(j = i; j \leq |V_p|; j{+}{+})$
      if $\big((avail(pm_j) == 0) \vee$ // pm_j is full
        $(i == j \wedge avail(pm_j) \leq 1)\big)$ **continue**;
      $c_{i,j} = 0$;
      $c(pm_i) = \mu \cdot c(p(v_k), pm_i)$, // cost of migrating v_k
      $c(pm_j) = \mu \cdot c(p(v_k^{'}), pm_j)$; // to pm_i, $v_k^{'}$ to pm_j
      $c_{i,j} = \lambda_k \cdot pr(i, j) + c(pm_i) + c(pm_j)$;
      if $(c_{i,j} < c_{min})$ $a = i$, $b = j$, $c_{min} = c_{i,j}$;
    end for;
  end for;
  $m = m \cup \{(pm_a, pm_b)\}$; // update migration scheme m
  $C_t(m, \overrightarrow{\pi}) \mathrel{+}= c_{min}$; // update total cost
  $avail(pm_a){-}{-}$, $avail(pm_b){-}{-}$;
  $k{+}{+}$; // the next VM pair
end while;
Migrate $(v_1, v_1{'})$, ..., $(v_l, v_l{'})$ according to m;
RETURN m and $C_t(m, \overrightarrow{\pi})$.

EXAMPLE 5: For the two VM pairs stored in the PADC of Fig. 3 (a), Algo. 4 will migrate both v_1 and $v_1^{'}$ to pm_1, resulting in cost of 406 for this pair. As pm_1 is now full, it will then migrate both v_2 and $v_2^{'}$ to pm_2, resulting in cost of 16 for this pair. The total cost of the two pairs is 422. □

Theorem 5: Under unordered policy, PAM is NP-hard even for one pair of VMs (i.e., $l = 1$).

Proof: We reduce a variation of the *s-t traveling salesman path problem* to this special case. By variation, we mean that in complete graph $K = (V_K, E_K)$, each node in V_K has a cost. Thus the cost of the *s-t* shortest Hamiltonian path includes the costs of s and t. The rest of the proof is then similar to that in Theorem 2 with one augmentation: For pm_i, its cost $c(pm_i)$ is the migration cost of v_1 from $p(v_1)$ to pm_i; for pm_j, its cost $c(pm_j)$ is the migration cost of $v_1{'}$ from $p(v_1^{'})$ to pm_j. ■

Theorem 6: Algo. 4 achieves 2-approximation when $l = 1$.

Proof: Let the pair of PMs that $(v_1, v_1^{'})$ migrate to computed by Algo. 4 be (pm_a, pm_b). Let the optimal VM migration of $(v_1, v_1^{'})$ be $(pm_{a'}, pm_{b'})$ and their optimal total cost be $C_t^{opt}(m, \overrightarrow{\pi})$. The total cost of $(v_1, v_1^{'})$ computed by Algo. 4 $C_c(m, \overrightarrow{\pi}) = \lambda_1 \cdot pr(a, b) + \mu \cdot c(p(v_1), pm_a) + \mu \cdot c(p(v_1^{'}), pm_b) \leq \lambda_1 \cdot pr(a', b') + \mu \cdot c(p(v_1), pm_{a'}) + \mu \cdot c(p(v_1^{'}), pm_{b'}) \leq 2 \cdot \lambda_1 \cdot opr(a', b') + 2 \cdot \mu \cdot c(p(v_1), pm_{a'}) + 2 \cdot \mu \cdot c(p(v_1^{'}), pm_{b'}) = 2 \cdot C_t^{opt}(m, \overrightarrow{\pi})$. ■

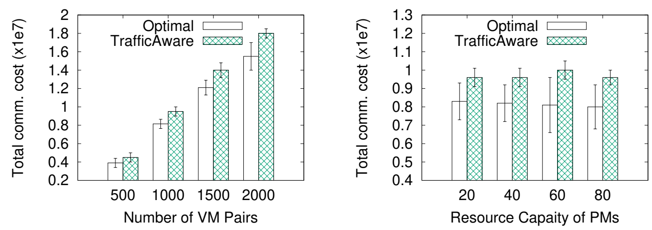
VI. Performance Evaluation

In this section we compare our PAL and PAM algorithms with existing work. For PAL, we name the optimal algorithm for ordered-policies (Algo. 1) as **Optimal** and the approximation algorithm for unordered (Algo. 3) as **Approx-PAL**, and compare them with **TrafficAware** [31], a popular VM placement algorithm that is policy-oblivious. For PAM, we refer to the minimum cost flow-based algorithm for ordered policies as **MCF** and the approximation algorithm for unordered (Algo. 4) as **Approx-PAM**, and compare them with **PLAN** [13].

TABLE II
Comparing PAM and PAL Algorithms.

	Ordered-Policy	Unordered-Policy	Existing Work
PAL	Optimal	Approx-PAL	TrafficAware [31]
PAM	MCF	Approx-PAM	PLAN [13]

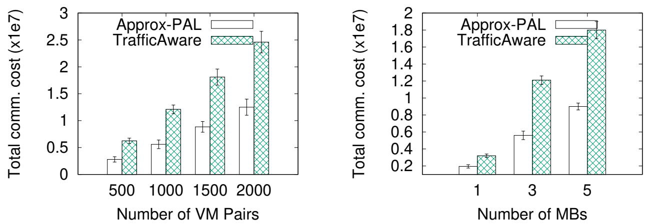
We consider fat-tree PADCs of size $k = 8$ with 128 PMs and size $k = 16$ with 1024 PMs. The traffic rates of VM pairs are in the range of [0, 1000] – Following flow characteristics found in Facebook data centers (Section 5.1, [34]), 25% of VM pairs have light traffic rates in [0, 300), 70% medium traffic rates in [300, 700], and 5% heavy rates in (700,1000]. As suggested by Cisco design guide [2], we install a number of MBs on different aggregation switches in the PADC. As 80% of cloud data center traffic originated by servers stays within the rack [7], for the initial VM placement in PAM, we place 80% of the VM pairs into the PMs under the same edge switches while the rest 20% under different edge switches. Each data point in the plots is an average of 20 runs with 95% confidence interval. In each run a new set of VM pairs are migrated (for PAM) or to be placed (for PAL) in the PADC.



(a) Varying l. $n = 3$, $rc = 40$. (b) Varying rc. $l = 1000$, $n = 3$.

Fig. 7. Comparing with TrafficAware in ordered-policy, $k = 16$.

Comparing with TrafficAware. As TrafficAware only assigns VMs (in the non-ascending order of their traffic rates) to the same PMs or PMs in close proximity, and does not consider the proximity of the PMs to the MBs, we implement TrafficAware as follows for fair comparison. In ordered-policy, it places VM pairs (in non-ascending order of their traffic rates) to the PMs that are closest to the ingress switch. In unordered policy, it works like Algo. 3 but only considers the Hamiltonian cycle case, as TrafficAware always places VM pairs in the same PM if possible. For ordered-policy, Fig. 7(a) varies the number of VM pairs l and shows that Optimal yields 15% less costs than TrafficAware. Fig. 7(b) varies resource capacities of PMs rc and shows that Optimal outperforms TrafficAware by around 15-20%. Fig. 8 compares Approx-PAL and TrafficAware under unordered-policy. It varies l as well as number of MBs n and shows that Approx-PAL outperforms TrafficAware by 40-50% in all scenarios. Above results are evident as Optimal and Approx-PAL are optimal and 2-approximation policy-aware algorithms while TrafficAware is policy-oblivious, inducing enormous traffic when VM communication traverses the MBs.



(a) Varying l. $n = 3$, $rc = 40$. (b) Varying n. $l = 1000$, $rc = 40$.

Fig. 8. Comparing with TrafficAware in unordered-policy, $k = 16$.

Effects of VM Migrations. We then investigate how much cost reduction VM migrations bring to a PADC ($k = 8$, $l = 1000$, $n = 3$) compared to without migrations. We consider that the VM migrations take place in *epochs*. At the beginning of each epoch, VM pairs change their traffic rates to new values in [0,1000] following aforesaid Facebook flow pattern. For migrations, it then executes MCF and calculates the total migration and communication costs. For non-migrations, it simply recalculates the total communication costs of VM pairs using the new traffic rates. We set the migration coefficient $\mu = 50$ and $\mu = 100$, and let the PADC run continuously for ten epochs. Fig. 9 shows the total cost of VM pairs in each epoch with and without VM migrations, for both $rc = 20$ and $rc = 40$. When $\mu = 100$, migration reduces the total costs of the VM pairs by around 10%, while when $\mu = 50$, it reduces by around 30%. In either case, cost for $rc = 40$ is smaller than $rc = 20$, as there are more resource slots available for cost-efficient VM migrations.

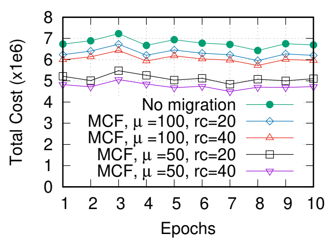
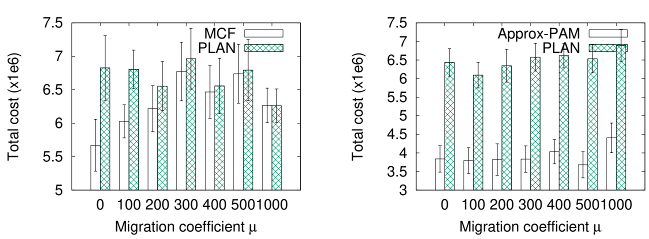


Fig. 9. Effects of VM migration.

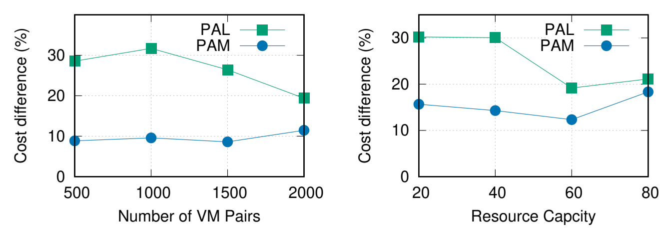


(a) Ordered Policy. (b) Unordered Policy.

Fig. 10. Comparing with PLAN, $k = 8$, $l = 1000$, $n = 3$, $rc = 20$.

Comparing with PLAN. We then compare our PAM algorithms with PLAN by increasing μ from 0 to 1000, the comparable range of the VM traffic rates. As PLAN is not designed for unordered policy, we implement it as below greedy algorithm. For each VM pair, PLAN finds a MB closest to the source VM as ingress MB and the one closest to the destination VM as egress MB. It then sets a MB sequence by starting from the ingress MB, visiting an unvisited closest MB, so on and so forth until all the MBs are visited and finally the egress MB is visited. Fig. 10(a) shows that when μ is small, the MCF outperforms the PLAN by around 20%. With the increase of μ, PLAN starts to perform close to MCF and finally the same at $\mu = 1000$ due to high migration cost. Fig. 10(b), however, shows that under unordered-policy, Approx-PAM outperforms PLAN by around 40% for the entire range of μ. Fig. 10(b) also shows that migration cost does not play a role in PLAN for unordered policy.

Comparing Ordered- and Unordered-Policy. Finally, we compare our own algorithms in ordered- and unordered-policy. For PAL algorithms, the cost difference is $\frac{\text{Optimal}-\text{Approx-PAL}}{\text{Optimal}}$; for PAM algorithms it's $\frac{\text{MCF}-\text{Appox-PAM}}{\text{MCF}}$. Fig. 11(a) shows that when varying l, Approx-PAL costs around 20-30% less than Optimal while Approx-PAM costs around 10% less than MCF. Fig. 11(b) shows that when varying rc, Approx-PAL costs around 20-30% less than Optimal while Approx-PAM costs around 10-20% less than MCF. This is because unlike in ordered policy wherein VMs must traverse the MBs in a specific order, in unordered policy, VM pairs can choose more cost-efficient sequences of MBs to traverse.



(a) Varying l. $n = 3$, $rc = 40$. (b) Varying rc. $l = 1000$, $n = 3$.

Fig. 11. Comparing ordered- and unordered-policy, $k = 16$, $\mu = 100$.

VII. Conclusions and Future Work

In this paper we study VM placement and VM migration in PADCs namely PAM and PAL. We uncover a suite of new policy-aware algorithmic problems and design optimal, approximation, and heuristic algorithms. We show VM migration is an effective technique to alleviate dynamic VM traffic in PADCs and our results outperform the state-of-the-arts. We will study if the optimality and approximability of our algorithms still hold when VMs have different resource demands. Finally, we will look into VNF migration and then a holistic VNF+VM migration to achieve ultimate resource optimization in PADCs.

Acknowledgment

This work was supported by NSF Grant CNS-1911191.

References

[1] Cisco global cloud index: Forecast and methodology, 2016 to 2021 white paper. https://www.cisco.com/c/en/us/solutions/service-provider/global-cloud-index-gci/white-paper-listing.html.

[2] Cisco virtualized multi-tenant data center, version 2.0 compact pod design guide. http://hyperurl.co/hpj2xt.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, 2008.

[5] M. Alicherry and T.V. Lakshman. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In *Proc. of IEEE INFOCOM*, 2013.

[6] A. Beloglazov and R. Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366 – 1379, 2013.

[7] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proc. of the ACM IMC 2010*.

[8] B. Carpenter and S. Brim. Middleboxes: Taxonomy and issues, 2002. https://tools.ietf.org/html/rfc3234.

[9] C. Clark, K. Fraser, and S. Hand. Live migration of virtual machines. In *Proc. of NSDI 2005*.

[10] R. Cohen, L. Lewin-Eytan, J. Seffi Naor, and D. Raz. Almost optimal virtual machine placement for traffic intense data centers. In *Proc. of Infocom 2013*.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.

[12] L. Cui, R. Cziva, F. P. Tso, and D. P. Pezaros. Synergistic policy and virtual machine consolidation in cloud data centers. In *Proc. of Infocom 2016*.

[13] L. Cui, F. P. Tso, D. P. Pezaros, W. Jia, and W. Zhao. Plan: Joint policy- and network-aware vm management for cloud data centers. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1163–1175, 2017.

[14] Y. Cui, Z. Yang, S. Xiao, X. Wang, and S. Yan. Traffic-aware virtual machine migration in topology-adaptive dcn. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 25(6):3427 – 3440, 2017.

[15] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca. An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *IEEE/ACM Transactions on Networking*, 25(4):2008–2025, 2017.

[16] J. Fan, C. Guan, Y. Zhao, and C. Qiao. Availability-aware mapping of service function chains. In *Proc. of IEEE INFOCOM 2017*.

[17] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In *Proc. of USENIX NSDI 2014*.

[18] G.R. Gupta G. Sallam, B. Li, and B. Ji. Shortest path and maximum flow problems under service function chaining constraints. In *Proc. of IEEE INFOCOM 2018*.

[19] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. Opennf: Enabling innovation in network function control. In *Proc. of the ACM SIGCOMM 2014*.

[20] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *J. Algorithms*, 22:1–29, 1997.

[21] A. Gushchin, A. Walid, and A. Tang. Scalable routing in sdn-enabled networks with consolidated middleboxes. In *Proc. of ACM Hotmiddlebox*, 2015.

[22] J.A. Hoogeveen. Analysis of christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, 10:291 – 295, 1991.

[23] H. Huang, S. Guo, J. Wu, and J. Li. Service chaining for hybrid network function. *IEEE Transactions on Cloud Computing*, 7:1082–1094, 2019.

[24] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *Proc. of the ACM SIGCOMM 2008*.

[25] T. Kuo, B. Liou, K. C. Lin, and M. Tsai. Deploying chains of virtual network functions: On the relation between link and server usage. *IEEE/ACM Transactions on Networking*, 26(4):1562–1576, Aug 2018.

[26] L. E. Li, V. Liaghat, H. Zhao, M. Hajiaghayi, D. Li, G. Wilfong, Y. R. Yang, and C. Guo. Pace: Policy-aware application cloud embedding. In *Proc. of IEEE INFOCOM 2013*.

[27] X. Li, J. Wu, S. Tang, and S. Lu. Let's stay together: Towards traffic aware virtual machine placement in data centers. In *Proc. of IEEE INFOCOM*, 2014.

[28] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin. Improve service chaining performance with optimized middlebox placement. *IEEE Transactions on Services Computing*, 10(4):560–573, 2017.

[29] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu. On dynamic service function chain deployment and readjustment. *IEEE Transactions on Network and Service Management*, 14(3):543–553, 2017.

[30] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer. Remedy: Network-aware steady state vm management for data centers. In *Proc. of the NETWORKING 2012*.

[31] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. of IEEE INFOCOM*, 2010.

[32] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood. Virtual function placement and traffic steering in flexible and dynamic software defined networks. In *The 2015 IEEE International Workshop on Local and Metropolitan Area Networks*.

[33] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simplefying middlebox policy enforcement using sdn. In *Proc. of the ACM SIGCOMM 2013*.

[34] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network's (datacenter) network. In *Proc. of the ACM SIGCOMM 2015*.

[35] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye. Provably efficient algorithms for joint placement and allocation of virtual network functions. In *Proc. of Infocom 2017*.

[36] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Proc. of NSDI 2012*.

[37] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: Network processing as a cloud servfce. In *Proc. of the ACM SIGCOMM 2012*.

[38] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.H. Liu, and S. Banerjee. Application-aware virtual machine migration in data centers. In *Proc. of Infocom 2011, mini conference*.

[39] H. Wang, Y. Li, Y. Zhang, and D. Jin. Virtual machine migration planning in software-defined networks. In *Proc. of Infocom 2015*.

[40] W. Wu Z. Xu J. Gu K. K. Ramakrishnan Y. He X. Qian Y. Jiang, Y. Cui. Speedybox: Low-latency nfv service chains with cross-nf runtime consolidation. In *Proc. of IEEE ICDCS 2019*.

[41] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan. Stochastic load balancing for virtual resource management in datacenters. *IEEE Transactions on Cloud Computing (Early Access)*, 2018.

[42] F. Zhang, G. Liu, X. Fu, and Ramin Yahyapour. A survey on virtual machine migration: Challenges, techniques, and open issues. *IEEE Communications Surveys & Tutorials*, 20:1206–1243, 2018.

[43] J. Zhang, F. Ren, and C. Lin. Delay guaranteed live migration of virtual machines. In *Proc. of Infocom 2014*.

[44] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patney, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula. Steering: A software-defined networking for inline service chaining. In *Proc. of the IEEE ICNP 2013*.